MIT GRAPHBASIERTER EDITION ZUR SEMANTISCHEN MULTIDIMENSIONALITÄT

Andreas Kuczera

Einleitung

Dieser Beitrag beschreibt Umfang und Funktionalität von "The Codex", einem Projekt, in dem die Möglichkeit zur mehrdimensionalen Annotation von Texten mit dem Management von Erschließungshierarchien vereint wird.[1] In relationalen oder XML-Datenbanken ist es oft schwer, Texte in ihrem direkten Kontext wahrzunehmen, da die narrative oder argumentative Struktur leicht verloren geht. Andererseits erleichtern sie statistische Auswertungen.

Auch XML ist ein leistungsstarkes Werkzeug für die Modellierung von Text, die beispielsweise mit XPATH ausgewertet werden können. Bei XML führt jedoch das Markup selbst zu Diskontinuitäten in Text und Annotationsdaten. Auch überlappende Annotationen, wie sie häufig bei der gleichzeitigen Auszeichnung von Layout und Semantik benötigt werden, können nicht direkt in der baumartigen hierarchischen Struktur von XML abgebildet werden. Dies erfordert Workarounds wie Standoff Markup, die wiederum die Lesbarkeit des XML-Dokuments beeinträchtigen.[2]

Codex zielt darauf ab, die Kluft zwischen überlappenden Annotationshierarchien und lesbarem Text zu überbrücken um eine „mehrdimensionale Integration" zu erreichen. Auf Markup wird völlig verzichtet. Stattdessen werden Standoff Properties zur Annotation verwendet. In den digitalen Geisteswissenschaften gibt es verschiedene Ansätze für Standoff Properties.[3] Codex verfolgt hier einen Ansatz, bei dem der Nutzer einen Text in Echtzeit annotieren kann und diese Annotationen frei überlappen können. In der Grundkonfiguration bietet Codex eine Auswahl von Stil-, Layout- und semantischen Annotationstypen. Es können auch Fußnoten, Marginalien usw. innerhalb des Editors hinzugefügt

1 Der vorliegende Beitrag ist eine überarbeitete Übersetzung eines Beitrages des Sonderbandes 4 der ZfdG, Die Modellierung des Zweifels – Schlüsselideen und -konzepte zur graphbasierten Modellierung von Unsicherheiten. Herausgegeben von Andreas Kuczera, Thorsten Wübbena, Thomas Kollatz mit dem Titel The Codex.net -- An Atlas of History, von Iian Neill and Andreas Kuczera.

2 Georg Vogeler, Patrick Sahle, XML, in: Fotis Jannidis/Hubertus Koh- le/Malte Rehbein (Hg.), Digital Humanities, Eine Einführung, Stuttgart 2017, S. 128-146, hier S. 134.

3 Zu Standoff-Markup im Rahmen der TEI vgl. https://wiki.tei-c.org/index.php/Stand-off_markup zuletzt abgerufen am 6.2.2019).

Abbildung 1. Stil-, Layout- und semantische Annotationstasten im Codex-Editor.

und annotiert werden. Annotationen selbst können in der browserbasierten, graphischen Oberfläche mit dem Erschließungsgraph verknüpft und selbst editiert und ergänzt werden.

Eines der Ziele von Codex ist es, über die Annotation von Entitäten und Ereignissen und deren Verknüpfung in den verschiedensten Quellen einen „Atlas von Beziehungen" zu erstellen. Dies ist mehr als eine Metapher für die Art der Annotationen und der Visualisierung ihrer Verbindungen: Atlas bezieht sich auf ein Netzwerk von Beziehungen, die auf verschiedene Weise projizierbar sind. Ziel ist es also nicht nur Entitäten im Text zu annotieren, sondern diese Entitäten als Knoten und Kanten in eine erschließende Graphstruktur einzubinden. Das Ergebnis ist sowohl ein annotiertes Dokument als auch eine Graphdatenbank, in der die Annotation in ihrem semantischen Kontext eingebettet ist.

Annotation

Es gibt momentan drei Hauptkategorien von Annotationen: Stil, Layout und Semantik. Beim Stil werden häufig verwendete typografische Stile wie z.B. Kursivschrift, Fettdruck, Unterstreichung, Durchstreichung, Hochstellung etc. angeboten. Dies ist aber komplett durch den Nutzer konfigurier- und erweiterbar.

Die interessanteren Kategorien sind jedoch Layout und Semantik. Diese gliedern sich wie folgt.

Layout

Seite, Absatz, Zeile, Satz, Spalte, etc.

Diese Annotationen beschreiben das Layout einer Seite in einem Manuskript. Da sich Standoff Properties frei überlappen können, stellen Überschneidungen, wie z.B. Absätze, die über einen Seitenwechsel hinweg reichen, kein Problem dar.

Trenn- und Bindestriche als Zero Point Annotation (ZPO)

Trenn- und Bindestriche stellen eine Herausforderung für die Kommentierung von Manuskripten dar; während der Editor die Position des Bindestrichs festhalten möchte, gibt es oft gute Gründe, Trennstriche bei der Weiterverarbeitung von Texten nicht zu berücksichtigen. Die Bindestrich-Annotation in Codex löst dieses Problem, indem sie die

Trennstriche als Zero Point Annotationen darstellt. Eine "nulldimensionale" Annotation ist ein Sonderfall einer Standoff-Property, die einen Anfangsindex, aber keinen Endindex hat. Auf diese Weise bezieht sich eine Annotation effektiv auf eine Position im Text zwischen den Zeichen. Der Bindestrich selbst wird nicht im Text gespeichert, die Worte bleiben unberührt. Im Editor wird die Annotation aber an der Position des Trennstrichs im Original in rot angezeigt. Zero Point Annotations sind ein verallgemeinerbares Merkmal von Standoff Properties und können auch für andere Fälle verwendet werden. Festzuhalten ist, dass beim Export ausgewählt werden kann, ob Trennstriche mit exportiert oder unterdrückt werden sollen.

Semantik

In Codex gibt es für jede semantische Annotation eine entsprechende semantische Einheit in der Graphdatenbank. Die verfügbaren Typen von Annotationen und Entitäten sind dabei frei konfigurierbar und werden nicht durch einen bestehenden Standard definiert. So kann das System in jedem Projektzusammenhang nach Wunsch konfiguriert werden. Jede Entität wird als eine Kombination von Knoten und Kanten in der Graphdatenbank modelliert. In Codex sind die für die Annotationen verwendeten Entitäten eigenständige Datensätze und sind damit unabhängig vom edierten Text.

Agents

Ein Agent bezieht sich auf jede Art von Entität, die im Text erwähnt wird. Dies können z.B. Personen, Orte, Objekte, Kollektive, Organisationen, Familien und andere Gruppen sein. Im Agentknoten selbst wird weiter differenziert, ob es sich um eine Person, einen Ort oder eine Gruppe etc. handelt. Metadaten über Agents sind ebenso in den Agentknoten gespeichert. Beispiele solcher Eigenschaften sind das Geschlecht einer Person, ihre Größe, ihr Gewicht etc.

Agents können auch über dynamisch erzeugte Beziehungen miteinander verbunden werden. (In Codex werden diese Meta-Relations genannt). Das folgende Beispiel zeigt einige der genealogischen Beziehungen von Lorenzo de' Medici, seine Verbindung zu Agents, die Kollektive darstellen sowie seine Anwesenheit in einer Gruppe von sechs florentinischen Botschaftern in Rom 1471.

RELATIONSHIP ID	VALUE	PROPERTY	TIME
3a79992e-ef90-4ecf-9f0c-6097931725bd	Male	Gender	

Abbildung 2. Das Geschlecht von Lorenzo de' Medici als Property.

RELATIONSHIP ID	RELATION	AGENT 2
2e0a3d13-8c5a-4762-903f-2d83df11ca2b	Part of	Six ambassadors [1471/09/23]
46a56b46-f962-4735-ba82-b7ff7dae6922	Brother of	Giuliano de' Medici
980feab9-ed6f-4b7d-b13d-0518333e1283	Part of	The Medici family
993428e1-efb9-43b3-a299-7ba23460f730	Child of	Cosimo de' Medici
cb1ad5d7-e02e-4393-8ca9-2ffa997b607e	Child of	Madonna Lucrezia [de' Medici]
fbc97384-90df-4a2e-8fbb-d0917cca45dd	Part of	Three Florentine ambassadors [1483/11/10]
2afb8754-1705-456a-ae32-97087991fe27	Married to	Madonna Clarice
00700f9b-5369-4a5e-a426-d1dead67d710	Parent of	Giovanni de' Medici

Abbildung 3. Eine Liste der Beziehungen von Lorenzo de‘ Medici.

Claim / Statement

Ein Claim bezieht sich auf eine Aussage über einen oder mehrere Agenten, in der Regel an einem Ort und zu einem Zeitpunkt. Ein Claim ist im Wesentlichen eine Aussage, die in der Regel im Rahmen eines Events erfolgt, aber auch einen Gedanken oder eine Meinung darstellen kann. Ein Claim in Codex wird nicht als Tatsachenaussage verstanden, sondern ist eine Datenstruktur, die einem RDF-Statement ähnelt. So ist beispielsweise die Aussage, dass „Lorenzo de‘ Medici auf seinem Anwesen in Careggi gestorben ist“, die Luca Landucci in seinem Tagebucheintrag vom 8. April 1492 gemacht hat, in Codex als

```
(Subject) Lorenzo de Medici
(Event) died
(At) Careggi
```

modelliert (Siehe Abbildungen 4 und 5). In Codex werden also “Tatsachen” als Statements gespeichert.

Texte

Eine Text-Einheit in Codex besteht aus Plaintext und einer Sammlung von Standoff-Properties. Der Einfachheit halber können Texte mit einem „Typ“ versehen werden, der

	ROLE	TYPE	NAME	AGENTS	TIME
View	Event	died	Lorenzo de' Medici died on his estate at Careggi	(According To) Luca Landucci (b.1436 - d. 1516; apothecary) (At) Careggi (Subject) Lorenzo de' Medici (b. 1449/01/01 - d. 1492/05/08; active from 1469/12/02)	On 1492/April/8 - -:--:--

Abbildung 4. Das Statement von Luca Landucci über den Tod von Lorenzo de‘ Medici in Codex.

ihre Funktion angibt (z.B. „`Body`“, „`Footnote`“, „`Marginnote`“, etc.). Dabei gibt es keine Einschränkungen hinsichtlich der Art des gespeicherten Textes. Im Falle von Luca Landuccis Tagebuch wird jeder Tagebucheintrag (von dem es mehrere pro Seite in der Quelle gibt) in einem separaten Textknoten gespeichert, während jeder Michelangelo-Brief als Ganzes in einem Textknoten gespeichert wird. Wird ein Text kommentiert, ist der Kommentartext wieder annotierbar, so dass der Kommentar selbst auch wieder annotiert werden kann. Die Annotationskette kann beliebig weitergeführt werden.

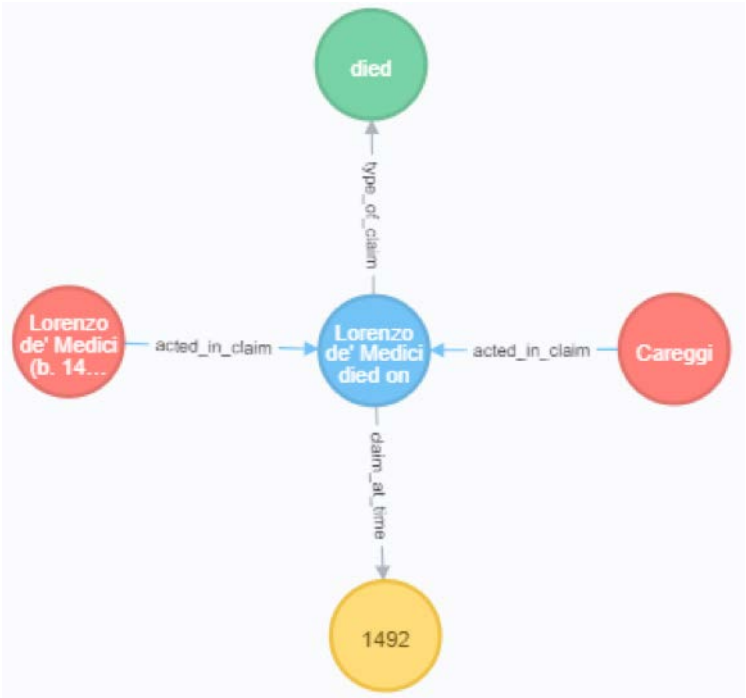

Abbildung 5. Eine Darstellung des obigen Statements als Knoten und Kanten im Neo4j-Browser. Der blaue Knoten ist ein Claim, die roten Knoten sind Agents, der grüne Knoten ist ein Concept und der gelbe Knoten ist die Zeitangabe.

Meta-Relations

Eine Meta-Relation ist eine Beziehung zwischen Agents mit zwei besonderen Merkmalen:

Meta-Relations sind in Codex dynamisch definierbar. Nach unseren Erfahrungen ist es von Vorteil Beziehungstypen anfangs nicht fest vorzugeben. Wenn der Benutzer hier frei ist, wird die spontane Erstellung von Beziehungstypen gefördert, man ist näher am Forschungszusammenhang. Dies schließt eine später Neu- und Umstrukturierung der Erschließungshierarchien nicht aus.

Meta-Relations sind bidirektional, d.h. der Benutzer kann beide Richtungen der Beziehung unterschiedlich benennen (z.B. Elternteil von / Kind von). Dies erleichtert das Denken in generischen Beziehung, z.B. „Abstammung". Ein Vorteil bei Cypher-Abfragen besteht darin, dass die Verbindung eines Agents zu Meta-Relation gefunden werden kann, ohne seine Rolle in der Beziehung kennen zu müssen.

Meta-Relations sind innerhalb einer Hierarchie modellierbar, so dass die Beziehungen selbst ein Graph sind. Beispielsweise kann man einen übergreifenden Beziehungstyp wie „zwischenmenschliche Beziehungen" definieren und darunter untergeordnete Typen wie „soziale Beziehungen", „Familienbeziehungen", „berufliche Beziehungen" usw. schachteln, so dass man Beziehungen zwischen Personen auf einer abstrakten Ebene abfragen kann. So kann eine Suche nach den „Freunden" einer Person leicht um „Mitarbeiter", „Bekannte" oder „Vertraute" ergänzt werden.

Eine Meta-Relation ist also eine Annotation, die in der Graphdatenbank durch einen Meta-Relation-Knoten dargestellt wird. Damit wird es möglich, Beziehungen von Agents in Texten zu kommentieren. In Abbildung 6 ist die orange Linie unter „Sohn von Antonio" eine Meta-Relation, die angibt, dass Luca Landucci der Sohn von Antonio Landucci war.

Concepts

Ein Concept in Codex ist eine Klasse oder ein Typ, der als Ganzes gesehen ein Konzept in der gemeinsamen Ontologie des Systems darstellt. Zu beachten ist, dass Ontologie hier nicht im Sinne einer „universellen" oder „Welt"-Ontologie zu verstehen ist, sondern lediglich als Subgraph, der von anderen Entitäten (wie Agents, Claims, Meta-Relations

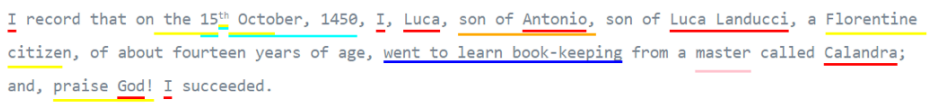

Abbildung 6. Beispiel für eine Meta-Relations-Annotation (orange unterlegt).

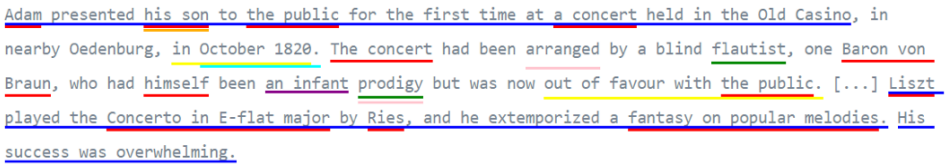

Abbildung 7. Konzeptkommentare (dunkelgrün) aus Alan Walkers Biographie von Franz Liszt.

usw.) im Sinne von gemeinsamen, wiederverwendbaren Konzepten geteilt wird. Anstatt eine universelle, top-down Ontologie zu verwenden, können die Nutzer sie bei Bedarf selbst definieren. Codex enthält bereits eine Reihe von Ontologien, wie z.B. Ontologien für Arten von Ereignissen, Orte, Beziehungen, Berufe. Das Concept ‚Event' ist der Wurzelknoten der Event-Ontologie und enthält alle Arten (und Subtypen) von Events, die in der Projektdomäne auftreten. Concepts können mehrere übergeordnete Elemente haben da ein Graph nicht an die Einschränkungen eines Baums gebunden ist. Die Änderung der Struktur der Ontologie (z.B. das Verschieben eines Kindkonzepts auf einen anderen Elternteil) ist in Codex einfach möglich, so dass ontologische Strukturen flexibel gehalten werden, um dem sich entwickelnden Verständnis der Projektdomäne gerecht werden zu können.

In Abbildung 7 stellen die dunkelgrünen Unterstriche das Konzept für „Flötist" und „Wunderkind" dar.

Datensätze und Datenpunkte

Ein Datenpunkt in Codex ist definiert als ein Maß in Raum und Zeit. Ein Datensatz ist wiederum eine Sammlung von Datenpunkten. Im folgenden Beispiel ist die Aussage, dass „drei Menschen an diesem Tag gestorben sind" als Datensatz der Datenpunkte „3", „Menschen", den Ort „Florenz" und die Zeit „23. April 1483" (Datum des Tagebucheintrags von Landucci) zusammenfasst.

Die Idee eines Datenpunktes besteht darin, es dem Editor zu ermöglichen, schnell numerische Daten aus einem Text zu extrahieren, die von statistischem Interesse sein können.

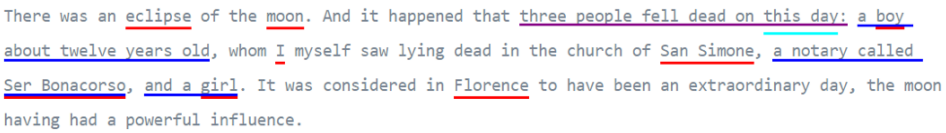

Abbildung 8. Datenpunkte (dunkelviolette Unterstreichung) in einem Tagebucheintrag von Landucci.

Abbildung 9. Das Eingabefenster für eine Zeiteinheit.

Einige praktische Beispiele für Datensätze, die aus historischen Quellen extrahiert werden können, sind epidemiologische Daten, Wetteraufzeichnungen, Volkszählungen, Kriminalitätsstatistiken usw.

Zeit

Die Modellierung von Zeitangaben ist in unterschiedlicher Genauigkeit möglich. Die Zeit-Entität besteht aus 9 Komponenten, die alle optional sind. Die möglichen Angaben umfassen `on`, `before`, `after`, `circa`, `early` und `late`. Die Jahreszeiten sind durch `Winter`, `Summer`, `Autumn`, `Spring` vertreten. Die Vielfalt der Optionen soll die Realitäten der Datumsangaben in historischen Texten widerspiegeln. Es ist geplant, die W3C Time Ontology als Grundlage hierfür zu nutzen.

In Codex ist es möglich, implizite Datumsangaben explizit zu machen. Die Zeitangabe in Abbildung 8 (Unterstreichung in türkis) mit dem Text „this day" wird mit dem angegebenen Datum des Tagebucheintrags (23. April 1483) verknüpft.

Nach dem Überblick zu den wichtigsten Annotationstypen wird nun das Standoff-Property-Modell vorgestellt. Es bildet die Grundlage um mit Codex Text-as-Graph editierbar zu machen.

Standoff-Properties

Eine Wortkette ist ein Graphmodell eines Textes, bei dem jedes Wort als Token-Knoten behandelt wird und die Serialität des Textes durch die Zuordnung jedes Knotens zu seinem Nachbarn mit einer NEXT_TOKEN-Kante modelliert wird.

Wie Standoff Properties stellt Text als Kette von Wortknoten eine markupfreie Alternative zu XML-Dokumentenformaten dar und und beherrschen die Modellierung von überlappenden Annotationshierarchien. Momentan fehlen noch einfache Nutzoberflächen für das direkte Management von Text-as-a-Graph-Modellen durch den Nutzer. Daher wird in Codex das Text-as-a-Graph-Konzept mit Hilfe eines Standoff-Property-Editors umgesetzt. Dies ist ein Kompromiss zwischen den multidimensionalen Möglichkeiten

Abbildung 10. Text als Kette von Wortknoten.

des Graphen und der technischen Robustheit und Nachhaltigkeit des Standoff-Property-Formats.

Die Entfernung von eingebettetem Markup macht den Textstrom sowohl für Menschen als auch für Maschinen leicht lesbar. Es löst auch das Problem der überlappenden Annotation, da die Eigenschaften getrennt vom Text gespeichert werden und keine hierarchischen Kodierungskonflikte entstehen. Mehrere Eigenschaften können sich über ihre Start- und Endindexe auf die gleichen Textbereiche oder überlappende Bereiche beziehen. Standoff-Properties sind von Natur aus diskrete Objekte, die in einer „flachen" Hierarchie koexistieren, d.h. ohne vorgeschriebene Hierarchie. Wird eine Annotation als Standoff-Property erstellt, kann sie unmittelbar mit dem Erschließungsgraphen verknüpft werden. Dies stellt die direkte Verbindung von Entitäten in der Graphdatenbank zu den entsprechenden Textbereichen her.

Eine Standoff-Property ist also im wesentlichen eine Datenstruktur mit folgenden Properties::

`Typ`: Eine Zeichenkette, die den Namen (d.h. den Typ) der Annotation angibt.

`StartIndex`: Eine ganze Zahl, die die Indexposition des ersten Zeichens der Annotation darstellt: 0 <= x < n, wobei x der Index ist und n die Länge des Textes ist.

`EndIndex`: Eine ganze Zahl, die die Indexposition des letzten Zeichens der Annotation innerhalb der Länge darstellt (gleiche Regel wie der StartIndex).

`Wert`: Eine Zeichenkette, die Daten darstellt, die spezifisch für die Annotation sind, wie beispielsweise die eindeutige Kennung einer referenzierten Entität, oder alternativ einen Farbwert, eine Textgröße, eine Schriftart usw.

`GUID`: Eine 32-stellige Zeichenkette, die als eindeutiger Identifikator für die Standoff-Property dient. Dies ist erforderlich, um sie in der Datenbank zu speichern.

`UserGUID`: Eine `GUID` (siehe oben), die den Benutzer repräsentiert, der die Annotation erstellt hat.

`Index`: Eine optionale ganze Zahl, die die Reihenfolge angibt, in der die Standoff-Propertys erstellt wurde.

`Text`: Eine optionale Zeichenkette, die den Quelltext darstellt, auf den sich die Annotation bezieht. Dies ist ein optionales Attribut, um die Anzeige von Standoff-Properties in der Datenbank zu erleichtern.

`Layer`: Eine optionale Zeichenkette, die beliebige Gruppen von Annotationstypen (Layer) repräsentiert (z.B. Layout, Struktur, Semantik).

`IsZeroPoint`: Ein boolescher Wert, der angibt, ob die Standoff-Property eine Zero-Point-Annotation ist.

`IsDeleted`: Ein boolescher Wert, der angibt, ob die Standoff-Eigenschaft als gelöscht markiert wurde.

Die bisher üblichen Standoff-Markup-Formate können in der Regel nicht mit der nachträglichen Änderung des Datums (des annotierten Textes) umgehen. Im Codex

Standoff-Property-Editor wird daher der Text als NodeList von HTML-SPAN-Elementen dargestellt, die über Referenzzeiger mit einem Array von JavaScript-Objekten mit den Standoff-Properties verbunden sind. Da eine verknüpfte Liste verwendet wird, um den Text darzustellen, und die Standoff-Properties mit dem Text mit Zeigern und nicht mit Indizes verknüpft sind, können Zeichen frei zu dem Text hinzugefügt oder entfernt werden, ohne dass Indexe während der Bearbeitung neu berechnet werden müssen. Erst wenn der Benutzer die Daten aus dem Editor exportiert (z.B. zum Speichern), werden die dynamischen Zeiger in statische Indexnummern umgewandelt. Der Plaintext und die Properties werden als JSON-Objekt exportiert, das in eine Textdatei gespeichert oder in ein Datenspeicherformat nach Wahl des Nutzers konvertiert werden kann. Codex übersetzt den JSON-Export in Standoff-Property-Knoten, die in einer Graphdatenbank gespeichert werden.

Beispiel

Nachfolgend ein Auszug aus dem JSON-Export des in Abb. 11 dargestellten Textes. Die gelben Hervorhebungen zeigen die typischen Teile der Standoff Properties. Die orangefarbenen Texte wiederholen den jeweils annotierten Text. Der Plaintext ist hellblau unterlegt.

Das volle Potenzial des Standoff-Property-Modells für die Integration von Text- und Graphstrukturen wird, abgesehen von der guten Lesbarkeit des Plaintextes ohne Mar-

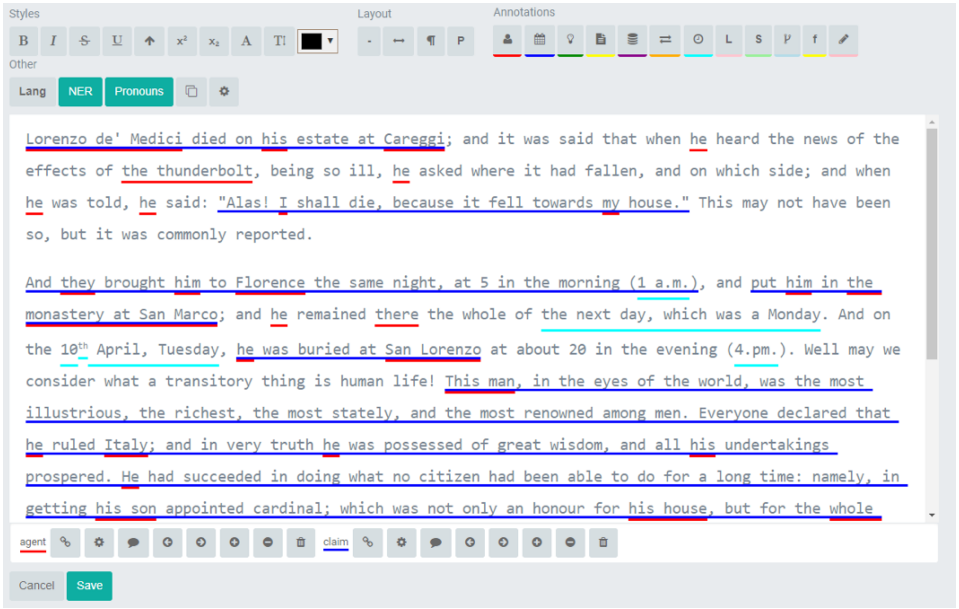

Abbildung 11. Luca Landucci's Tagebucheintrag vom 8. April 1492 über den Tod von Lorenzo de' Medici.

```
{
 “text”: “Lorenzo de’ Medici died on his estate at Careggi; and it was
said that when he heard the news of the effects of the thunderbolt,
being so ill, he asked where it had fallen, and on which side; ... “,
 “properties”: [{
       “index”: 23,
       “guid”: “cde24f38-81cb-4110-b368-b5b1f4ed4d53”,
       “type”: “agent”,
       “layer”: null,
       “text”: “Lorenzo de’ Medici”,
       “value”: “e45fed44-17a0-4c2c-9c00-858667a17904”,
       “startIndex”: 0,
       “endIndex”: 17,
       “isZeroPoint”: false,
       “isDeleted”: false,
       “userGuid”: “fb067f75-a121-47c1-8767-99271c75cfc0”
 }, {
       “index”: 25,
       “guid”: “5e33d2a8-dea0-4bbf-b4f6-8ccf40dac4d9”,
       “type”: “agent”,
       “layer”: null,
       “text”: “Careggi”,
       “value”: “d8eda97c-79d7-43ad-b43f-fd3e3a05c68e”,
       “startIndex”: 41,
       “endIndex”: 47,
       “isZeroPoint”: false,
       “isDeleted”: false,
       “userGuid”: “fb067f75-a121-47c1-8767-99271c75cfc0”
 }, {
       “index”: 46,
       “guid”: “94eebe67-0136-4f54-a4c6-6e7072dfb3de”,
       “type”: “claim”,
       “layer”: null,
       “text”: “Lorenzo de’ Medici died on his estate at Careggi”,
       “value”: “175742e2-a342-455d-a1b9-3fe3a96e3fd9”,
       “startIndex”: 0,
       “endIndex”: 47,
       “isZeroPoint”: false,
       “isDeleted”: false,
       “userGuid”: “fb067f75-a121-47c1-8767-99271c75cfc0”
 }]
}
```


Abbildung 12. Auszug aus dem JSON-Export von Landucci‘s Tagebucheintrag.

kup, den überlappenden Annotationen mit ihren direkten Verknüpfungen zu präzisen Textstellen vor allem durch zwei Merkmale klar:

Standoff-Properties können in sogenannten Layern gruppiert werden, wobei ein Layer entweder implizit durch den Annotationstyp oder explizit durch einen im Layer-Attribut gespeicherten Wert definiert wird.

Die Attribute Startindex und Endindex bieten die Möglichkeit, Annotationen zu kombinieren, die im gleichen Textbereich enthalten sind oder sich überlappen.

Diese Möglichkeiten zur Schichtung und Kombination haben bereits zu interessanten Funktionen im Codex-Editor geführt (z.B. bei der Verwaltung von Named Entities und Pronomen-Annotationen). Sie bieten aber auch weitere Erkenntnisperspektiven. So wird es möglich über eine Cypher-Abfrage alle Annotationen in einem Text (oder in allen Texten) zu finden, die sich überlappen. Kombinationen von Annotationen können den von ihnen kommentierten Text wiederum mit neuen Bedeutungen anreichern.

Schließlich ist es mit Standoff-Properties möglich, verschiedene Annotationskonzepte in einem System zu vereinen, z.B. Layout, Semantik und Syntaxanalyse.

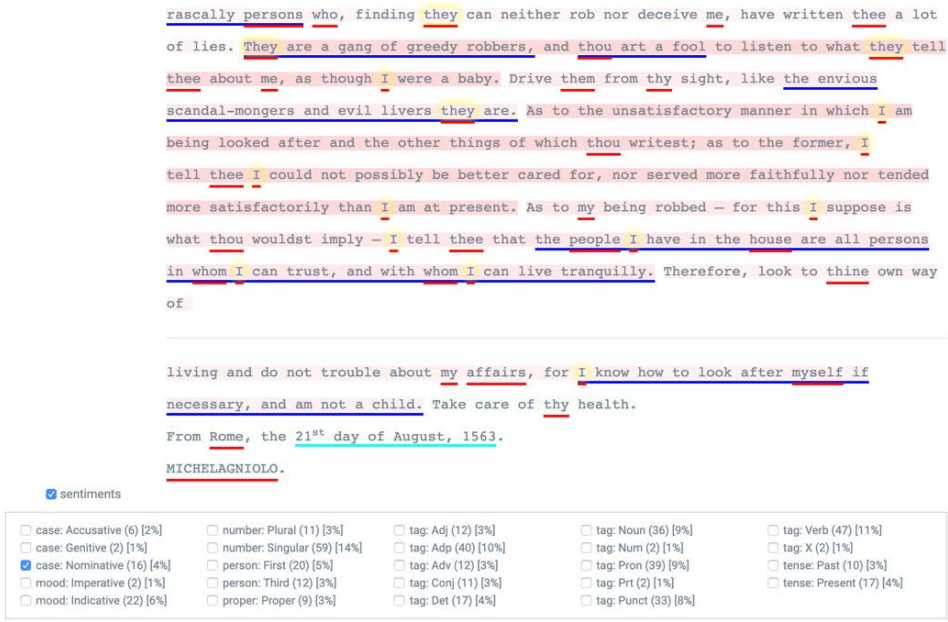

Abb. 13: Visualisierung der Syntax- und Sentimentanalyse auf Grundlage von Standoff Properties.

Weitere Annotationsebenen

In der folgenden Abbildung werden Sentiment- und Syntaxanalyse für einen Michelangelo-Brief dargestellt. Im Rahmen des hier beantragten Projekts werden auch Annotationen zu Syntax, Schreiberhänden, Sprache und Sprachgebrauch etc. denkbar.

Im folgenden werden die technischen Grundlagen von SPEEDy (Editorkomponente) und des Codex-Systems rund um den Editor beschrieben. SPEEDy dient dabei zur direkten Transkription von Text. In Codex werden die Annotationsinformationen strukturiert gespeichert und damit auch erschlossen.

Fazit

Codex verwendet Standoff Properties, um Interpretation und Text im Graph zu vereinen. Annotationen werden dem Text auf Zeichenebene zugeordnet und können uneingeschränkt überlagert werden. Mit kommentierten Annotationen ergeben sich sogar Perspektiven, den wissenschaftlichen Diskurs nachvollziehbar zu machen. Dabei kann jede Annotationen - einschließlich Events, Meta-Relationen, Agents etc. - jederzeit auf ihre textliche Grundlage zurückgeführt werden.